

A large, stylized letter 'A' is formed using the characters 'S' and 'Y'. The 'S' characters are arranged in a grid-like pattern to form the left and right sides of the letter, while 'Y' characters form the central vertical stem and the horizontal crossbar. The overall shape is a bold, blocky 'A' that fills most of the page.

[illegible]

(2)	102
(3)	178
(4)	485
(5)	597
(6)	687
(7)	794

DECLARATIONS
SYSSETPFM - Initialize Page Fault Monitoring
PFM\$PURGE - Return all process buffers to pool
PFM\$GETBUF - Return PFM Buffer to Caller
ALLPMB - Allocate PMB control block and data buffers
PFM\$MON - Resident Monitoring Code


```
0000 1      .TITLE SYSSETPFM - SET PAGE FAULT MONITORING
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 *****
0000 6
0000 7      * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      * ALL RIGHTS RESERVED.
0000 10
0000 11      * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12      * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13      * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14      * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15      * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16      * TRANSFERRED.
0000 17
0000 18      * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19      * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20      * CORPORATION.
0000 21
0000 22      * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23      * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26
0000 27
0000 28      **
0000 29      FACILITY: Measurement System Service
0000 30
0000 31      ABSTRACT:
0000 32
0000 33      This module enables a page fault monitoring service within the
0000 34      operating system. On each page fault, the virtual address, the PC,
0000 35      and the process CPU time are saved in a buffer to be output by either
0000 36      a cooperating subprocess or an image-based AST routine.
0000 37
0000 38      ENVIRONMENT: Kernel Mode
0000 39
0000 40      AUTHOR: Henry M. Levy , CREATION DATE: 7-May-1977
0000 41
0000 42      MODIFIED BY:
0000 43
0000 44      V03-007 SSA0026      Stan Amway      9-Jul-1984
0000 45      In PFMSMON, raise IPL to IPL$ HWCLK when
0000 46      doing CPU time reference update.
0000 47
0000 48      V03-006 SSA0025      Stan Amway      25-Jun-1984
0000 49      Add global symbol PFMS$ _BUFCNT for use by PCA.
0000 50
0000 51      V03-005 SSA0019      Stan Amway      12-Mar-1984
0000 52      Prevent subprocess running PFMFILWRT from deallocating PMB.
0000 53      Add access mode checking for subfunction and stop requests.
0000 54
0000 55      V03-004 SSA0013      Stan Amway      28-Feb-1984
0000 56      Properly handle monitoring termination by
0000 57      dequeuing AST block from PCB.
```



```
0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :--
```

Ensure that PFMSPURGE will be invoked if service is called with a stop monitoring request (notably from SYSRUNDWN) and a PMB is still allocated. Inhibit buffer flush if buffer contains no records. Fix bug that inhibited the entry of a CPU timestamp into the first page fault buffer. Move PFMSGETBUF and ALLPMB routines to paged PSECT.

V03-003 SSA0007 Stan Amway 2-Feb-1984
Removed restriction of 1 process per group. Added AST interlock flag to keep the number of ASTs delivered to a minimum. Track changes in buffer format.

V03-002 SSA0004 Stan Amway 12-Dec-1983
Extensive changes to add support for the Performance & Coverage Analyzer (PCA) being done by the VMS Debug group.

Changes include:
Removed use of PMB list. Use PCB\$PMB (new) instead. Timestamping fault with process CPU time. Adding support for image-based buffer handling. Cleanup of error handling. Optimizing main code paths for speed.

V03-001 CWH1002 CW Hobbs 1-Mar-1983
Convert to use extended pids.

02 RIH0033 R. I. HUSTVEDT 16-OCT-1979
CHANGE PCB\$W_BYTCNT TO JIB\$L_BYTCNT.

03 BLS0001 B. L. SCHREIBER 28-NOV-1979
CORRECT PAGE FAULT ERROR

04 BLS0002 B.L. SCHREIBER 28-JAN-1980
CORRECT ERROR IN PURGE ROUTINE.

05 BLS0003 B.L. SCHREIBER 30-JAN-1980
MORE ERRORS IN PURGE ROUTINE


```
0000 102      .SBTTL  DECLARATIONS
0000 103      :
0000 104      : INCLUDE FILES:
0000 105      :
0000 106      :
0000 107      :
0000 108      : MACROS:
0000 109      :
0000 110      :
0000 111      $ACBDEF      ; define AST control block
0000 112      $DYNDEF      ; define dynamic structure types
0000 113      $IODEF       ; define I/O function codes
0000 114      $IPLDEF      ; define interrupt priority levels
0000 115      $JIBDEF      ; define job information block
0000 116      $PCBDEF      ; define process control block
0000 117      $PHDDEF      ; define process header
0000 118      $PFBDEF      ; define PFM buffer layout
0000 119      $PMBDEF      ; define PFM control block
0000 120      $PQLDEF      ; define process quota codes
0000 121      $PRDEF       ; define processor registers
0000 122      $PRIDEF      ; define priority increment classes
0000 123      $PSLDEF      ; define PSL fields
0000 124      $SGNDEF      ; define system parameters
0000 125      $SSDEF       ; define service status codes
0000 126      :
0000 127      .MACRO  $QUOTA  NAME=LISTEND,VALUE=0
0000 128      .BYTE  PQL$ 'NAME
0000 129      .LONG  VALUE
0000 130      .ENDM  $QUOTA
0000 131      :
0000 132      :
0000 133      : EQUATED SYMBOLS:
0000 134      :
0000 135      :
00000004 0000 136 PFMFLG = 4      ; Argument list offsets
00000008 0000 137 ASTADR = 8      ; Function/subfunction flags
0000000C 0000 138 ASTPRM = 12     ; AST Routine Address
00000010 0000 139 ACMODE = 16     ; AST Parameter
0000 140      :
00000005 0000 141 BUFCNT = 5      ; number of buffers to allocate
00000005 0000 142 PFM$C BUFCNT == BUFCNT ; global definition of BUFCNT
0000020C 0000 143 BUFSIZ = PFB$C LENGTH ; size of buffers
0000003F 0000 144 MAXREC = <<BUFSIZ-PFB$B BUFFER>/<2*4>> ; max records per buffer
00000A7C 0000 145 QUOTA CHARGE = <<BUFCNT*BUFSIZ> + PMB$C_LENGTH> ; Amt to charge process byte quota
0000003C 0000 146 FAULTVA = <4*7>+4+<7*4> ; offset to va of faulting instruction
00000040 0000 147 FAULTPC = <4*7>+8+<7*4> ; offset to pc of faulting instruction
0000 148      :
0000 149      :
0000 150      : OWN STORAGE:
0000 151      :
0000 152      :
00000000 0000 153      .PSECT  YEXEPAGED, LONG
0000 154      :
0000 155      :
0000 156      : Data for creation of subprocess to output filled
0000 157      : buffers to disk file.
0000 158      :
```



```
0000 159
0000 160 PFMFILWRT: ; subprocess image name descriptor
0000 161 .LONG 20$-10$
0004 162 .LONG 10$
50 3A 4D 45 54 53 59 53 24 53 59 53 0008 163 10$: .ASCII /SYS$SYSTEM:PFMFILWRT.EXE/
45 58 45 2E 54 52 57 4C 49 46 4D 46 0014
0020 164 20$:
0020 165 PFMQUOTA: ; subprocess quota name
0020 166 $QUOTA CPULM,0 ; infinite CPU time
0025 167 $QUOTA BYTLM,1024 ; byte limit for buffered I/O
002A 168 $QUOTA FILLM,1 ; open file count limit
002F 169 $QUOTA PGFLQUOTA,256 ; paging file quota
0034 170 $QUOTA PRCLM,0 ; no subprocesses
0039 171 $QUOTA TQELM,1 ; timer queue entry
003E 172 $QUOTA LISTEND ; end of list
49 50 45 20 3C 5F 42 55 53 4D 46 50 0043 173 FILWRT: ; subprocess process name
3E 20 44 0043 174 .ASCII /PFMSUB_< EPID >/
004F
0052 175 FILWRTPRV: ; subprocess privilege vector
FFFFFF FFFFFFFF 0052 176 .LONG -1,-1 ; all privileges
```

```
005A 178 .SBTTL SYSSETPFM - Initialize Page Fault Monitoring
005A 179 :++
005A 180 : FUNCTIONAL DESCRIPTION:
005A 181 :
005A 182 : Page fault monitoring initialization. Buffers are allocated from
005A 183 : the nonpaged pool and queued for use by the monitor. In subprocess
005A 184 : mode, a subprocess is created which outputs buffers which have been
005A 185 : filled. In image-based mode, ASTs are delivered to signal full buffers.
005A 186 :
005A 187 : When the process calls SETPFM to turn off monitoring, all
005A 188 : buffers are returned to the system and if in subprocess mode, the
005A 189 : subprocess is deleted.
005A 190 :
005A 191 : In case of abnormal termination, the buffers are returned by SYSRUNDWN.
005A 192 :
005A 193 : CALLING SEQUENCE:
005A 194 :
005A 195 : CALLS/CALLG
005A 196 :
005A 197 : INPUT PARAMETERS:
005A 198 :
005A 199 : 4(AP) PFMFLG Function/subfunction
005A 200 : bit 0 = off/on (0/1)
005A 201 : bits 1-30 = subfunction field
005A 202 : bit 1 = Flush buffers
005A 203 : bit 31 = 0 indicates initialization call
005A 204 : 1 indicates subfunction call (if bit 0 = 1)
005A 205 :
005A 206 : 8(AP) ASTADR AST Routine Address
005A 207 : = 0 Subprocess Mode
005A 208 : <> 0 AST Routine address for image-based buffer handler
005A 209 :
005A 210 : 12(AP) ASTPRM AST parameter
005A 211 :
005A 212 : 16(AP) A(MODE) Access Mode for AST delivery
005A 213 :
005A 214 :
005A 215 : IMPLICIT INPUTS:
005A 216 :
005A 217 : none
005A 218 :
005A 219 : OUTPUT PARAMETERS:
005A 220 :
005A 221 : none
005A 222 :
005A 223 : IMPLICIT OUTPUTS:
005A 224 :
005A 225 : none
005A 226 :
005A 227 : COMPLETION CODES:
005A 228 :
005A 229 : SSS_NORMAL - Success
005A 230 : SSS_EXQUOTA - A quota was exceeded while allocating buffers
005A 231 : or creating the cooperating subprocess.
005A 232 : SSS_INSFMEM - Insufficient dynamic memory was available for buffering
005A 233 : SSS_PFMBSY - Attempted to initialize page fault monitoring while
005A 234 : already active.
```



```
005A 235 : SSS_NOPRIV - Caller's access mode is less privileged than the mode
005A 236 : that started page fault monitoring.
005A 237 :
005A 238 : SIDE EFFECTS:
005A 239 :
005A 240 : none
005A 241 :
005A 242 : --
005A 243 :
005A 244 :
005A 245 :
0000005A 246 : .PSECT YEXEPAGED
005A 247 :
005A 248 : .ENABLE LSB
005A 249 PFMBSY:
50 0204 8F 3C 005A 250 MOVZWL #SS$_PFMBSY,R0
05 11 005F 251 BRB 10$
0061 252 ILLSEQOP:
50 02DC 8F 3C 0061 253 MOVZWL #SS$_ILLSEQOP,R0
0066 254 10$:
0066 255 : .DISABLE LSB
0066 256 SETIPL #0 ; Restore IPL
0069 257 RET ; Return w/error in R0
006A 258
OFFC 006A 259 : .ENTRY EXE$SETPFM,*M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
006C 260
54 00000000'EF D0 006C 261 MOVL L^SCH$GL_CURPCB,R4 ; get our PCB address
55 00000000'9F D0 0073 262 MOVL @#CTL$GL-PHD,R5 ; and PHD address (P1 window)
007A 263 SETIPL #IPL$_ASTDEL ; Protect access to PMB
56 011C C4 D0 007D 264 MOVL PCB$_PMB(R4),R6 ; Get address of allocated PMB (if any)
53 04 AC D0 0082 265 MOVL PFMFLG(AP),R3 ; Get copy of PFMFLG
0086 266
0086 267 : From this point on, IPL is at IPL$_ASTDEL and registers are as follows:
0086 268 :
0086 269 : R3 Copy of PFMFLG
0086 270 : R4 PCB Address
0086 271 : R5 PHD Address (P1 window)
0086 272 : R6 PMB Address or 0 (initialize request only)
0086 273 :
0086 274 :
0086 275 : *** Order of following tests makes
0086 276 : implicit checks on PFM state ***
58 36 A5 E1 0086 277 BBC #PHD$V_PFMFLG,- ; BR if monitoring not initialized
37 53 E9 0088 278 PHD$V_FLAGS(R5),START
D1 18 008E 279 BLBC R3,STOP_VEC ; BR if termination request
0090 280 BGEQ ILLSEQOP ; BR if R3 >= 0 (not a subfunction call)
0090 281 :
0090 282 : Subfunction Processing
0090 283 :
0090 284 :
0090 285 SUBFUNC:
0090 286 BSBB CHECK_ACMODE ; If caller does not have privilege,
0092 287 BLBC R0,NOPRIV ; return with error status
0095 288 BBC #1,R3,10$ ; If flush buffer request,
0099 289 JSB FLUSH_BUFFER ; do it now
009F 290 ; (R2, R4, R5 DESTROYED)
009F 291 ASSUME PMB$V_MODE EQ 0
```

```
02 0B A6 E9 009F 292 ASSUME PMBSK_IMAGE EQ 1
71 10 009F 293 10$: BLBC PMBSB_FLAGS(R6),20$ ; If image-based mode,
00A3 294 BSBB SET_ASTMODE ; update AST parameters
00A5 295 20$: SETIPL #0 ; Lower IPL
50 01 3C 00A8 296 MOVZWL #SS$_NORMAL,R0 ; and return with success
04 00AB 297 RET
00AC 298
00AC 299 ; Determine if caller is
00AC 300 ; permitted to execute function
00AC 301 CHECK_ACMODE:
50 D4 00AC 302 CLRL R0 ; Assume function not allowed to caller
51 DC 00AE 303 MOVPSL R1 ; R1 <= PSL
02 16 EF 00B0 304 EXTZV #PSL$V_PRVMOD,#PSL$S_PRVMOD,- ; R1 <= caller's mode
51 51 00B3 305 R1,R1
2E A6 51 91 00B5 306 CMPB R1,PMBSB_ACMODE(R6) ; If caller less privileged than owner,
02 1A 00B9 307 BGTRU 10$ ; return with no access indication in R0
50 D6 00BB 308 INCL R0 ; else return access OK
05 00BD 309 10$: RSB
00BE 310
00BE 311 NOPRIV:
50 24 3C 00BE 312 MOVZWL #SS$_NOPRIV,R0 ; Return w/no privilege error
04 00C1 313 SETIPL #0
00C4 314 RET
00C5 315
00C5 316 STOP_VEC:
0142 31 BRW STOP
00C8 318 PURGE_EXIT RO_VEC:
0161 31 BRQ PURGE_EXIT_RO
00CB 320 PURGE_EXIT_VEC:
015B 31 BRQ PURGE_EXIT
00CE 321
00CE 322
00CE 323 CHECK_STOP:
90 53 E8 00CE 324 BLBS R3,ILLSEQOP ; Error if not termination request
56 D5 00D1 325 TSTL R6 ; If PMB doesn't exist,
8C 13 00D3 326 BEQL ILLSEQOP ; don't proceed
00 E0 00D5 327 BBS #PMB$V_MODE,- ; If image mode, continue normally
EB 0B A6 00D7 328 PMBSB_FLAGS(R6),STOP_VEC
34 A6 D1 00DA 329 CMPL PMBSL_EPID(R6),- ; Don't allow PFM rundown
64 A4 00DD 330 PCBSL_EPID(R4) ; if the current process is the
80 13 00DF 331 BEQL ILLSEQOP ; subprocess running PFMFILWRT
E2 11 00E1 332 BRB STOP_VEC ; Otherwise, join STOP monitoring code
00E3 333
00E3 334 Initialize page fault monitoring
00E3 335
00E3 336
00E3 337 START:
01 53 D1 00E3 338 CMPL R3,#1 ; Valid request ?
E6 12 00E6 339 BNEQ CHECK_STOP ; BR if not
56 D5 00E8 340 TSTL R6 ; PMB allocated ?
03 13 00EA 341 BEQL 10$
FF6B 31 00EC 342 BRW PFMBUSY ; Yes - can't proceed
0249 30 00EF 343 10$: BSBW ALLPMB ; Allocate PMB
00F2 344 SETIPL #0 ; It is now safe to lower IPL
D0 50 E9 00F5 345 BLBC R0,PURGE_EXIT_RO_VEC ; If PMB allocation error, quit now
51 DC 00F8 346 MOVPSL R1 ; Store caller's access mode
02 16 EF 00FA 347 EXTZV #PSL$V_PRVMOD,#PSL$S_PRVMOD,-
51 51 00FD 348 R1,R1 ; as owner's access mode
```



```
2E A6 51 90 00FF 349      MOVB    R1,PMB$B ACMODE(R6)      ; for subsequent privilege checking
      08 AC D5 0103 350      TSTL    ASTADR(AP)          ; Are we initializing image-based mode ?
      28 13 0106 351      BEQL     SET_SUBPMODE         ; If no, setup subprocess mode
      OC 10 0108 352      BSBB     SET_ASTMODE           ; Else setup image-based AST mode
      010A 353      ASSUME    PMB$V_MODE EQ 0
      010A 354      ASSUME    PMB$K_IMAGE EQ 1
      0B A6 96 010A 355      INCB     PMB$B_FLAGS(R6)      ; Set image-based AST mode flag
      010D 356      START_OK:
      00 00 E2 010D 357      BBSS     #PHD$V_PFMFLG,-      ; Enable page fault monitoring
00 36 A5 010F 358      10$:      PHD$W_FLAGS(R5),10$
50 01 3C 0112 359      MOVZWL    #SS$ _NORMAL,R0          ; and return with success status
      04 0115 360      RET
      0116 361
      0116 362
      0116 363      ;
      0116 364      ;      Setup image-based mode AST parameters
      0116 365
      0116 366      SET_ASTMODE:
      0116 367      ASSUME    PMB$L_ASTPRM EQ PMB$L_AST+4
      0116 368      ASSUME    ASTPRM EQ ASTADR+4
      34 A6 08 AC 7D 0116 369      MOVQ     ASTADR(AP),PMB$L_AST(R6); Save AST address and parameter
50 10 AC 02 00 EF 011B 370      EXTZV    #0,#2,ACMODE(AP),R0 ; R0 <= request AST delivery mode
      FEDC' 30 0121 371      BSBW     EXE$MAXACMODE         ; Maximize requested and allowable
      30 89 0124 372      BISB3     #<ACB$M NODELETE!ACB$M_PKAST>,- ; access modes & store with
      2F A6 50 0126 373      R0,PMB$B_RMOD(R6) ; nodelete, pkast set;quota, kast, clear
3C A6 02D1'CF 9E 0129 374      MOVAB     W^PFM_PKAST,PMB$L_KAST(R6) ; Set piggy-back kernel AST address
      05 012F 375      RSB
      0130 376
      0130 377      ;
      0130 378      ;      Setup subprocess mode
      0130 379
      0130 380
      0130 381      SET_SUBPMODE:
      0130 382      ; (R0-R3 DESTROYED)
      01  DD 0130 383      PUSHL     #SS$ _NORMAL          ; Save room for status on stack
      0132 384      ;
      0132 385      ;      Create a termination mailbox for the subprocess.
      0132 386
      0132 387
      0132 388      $CREMBX_S CHAN=PMB$W_MBXCHN(R6),MAXMSG=#120,-
      0132 389      -BUFQUO=#120,PROMSK=#0
      014E 390
      6E 50 D0 014E 391      MOVL     R0,(SP)
      03 50 E8 0151 392      BLBS     R0,5$
      00D2 31 0154 393      BRW      PURGE_EXIT
      5E 10 C2 0157 394      5$:      SUBL2     #16,SP      ; exit on error
      5E DD 015A 395      PUSHL     SP                  ; buffer space for GETCHN on stack
      10 DD 015C 396      PUSHL     #16                  ; build descriptor for buffer
      52 6E DE 015E 397      MOVAL     (SP),R2           ; length of buffer
      0161 398      $GETCHN_S CHAN=PMB$W_MBXCHN(R6),PRIBUF=(R2) ; get descriptor address
      0174 399      ; get mailbox unit #
      0174 400
      0174 401      ; Form unique subprocess name using EPID of this process
      0174 402      ; (NOTE: The following code adds a NET 4 bytes to the stack local storage)
      0174 403
      0174 404
      3C BB 0174 405      PUSHR     #^M<R2,R3,R4,R5>
```

```
04 A2  FEC8 CF  0F 28 0176 406      MOV C3 #15,W^FILWRT,4(R2)      ; Move name template to stack storage
          3C  BA 017D 407      POPR    #^M<R2,R3,R4,R5>
          51 64 A4 D0 017F 408      ; Convert hex EPID to ASCII
          57 0B A2 9E 017F 409      ; R1 <= EPID to be converted
          58 1C D0 0183 410      ; R7 <= address of 1st character
          50 51 04 58 EF 0187 411      ; R8 <= position of 1st nibble
          87 00000000'9F 40 90 018A 412 10$: EXTZV R8,#4,R1,R0      ; Get next four bits of value
          FFEC 58 FC 8F 00 9D 018F 413      ; Convert to ASCII and store
          62 04 A2 9E 0197 414      ; Loop until all digits converted
          62 04 A2 9E 019E 415      ; Form string descriptor
          0F DD 01A2 416      PUSHL    #15      ; for resultant name
          01A4 417      ;
          01A4 418      ; Create subprocess with high priority, full privilege and termination mailbox
          01A4 419      ;
          01A4 420      ;
          01A4 421      ;
          01A4 422      ;
          01A4 423      ;
          01A4 424      ;
          01A4 425      ;
          01A4 426      ;
          01A4 427      ;
          01A4 428      ;
          5E 1C C0 01CF 429      ADDL2    #<24+4>,SP      ; Return stack local storage
          6E 50 D0 01D2 430      MOVL     R0,(SP)
          03 50 E8 01D5 431      BLBS     R0,15$
          0082 31 01D8 432      BRW      DASSGN EXIT      ; exit on error
          7E 54 7D 01DB 433 15$: MOVQ     R4,-(SP)      ; Save R4, R5
          50 34 A6 D0 01E2 434      SETIPL  B^20$      ; Synchronize access to system database
          00000000'EF 16 01E6 435      MOVL     PMBSL_EPID(R6),R0      ; Convert EPID to IPID
          30 A6 50 D0 01EC 436      JSB      EXE$EPID_TO_IPID
          00000000'EF 16 01F0 437      MOVL     R0,PMBSL_PID(R6)      ; and save in PMB
          011C C0 56 D0 01F6 438      JSB      EXE$IPID_TO_PCB      ; Convert IPID to PCB address
          54 8E 7D 01FB 439      MOVL     R6,PCBSL_PMB(R0)      ; Insert PMB address in subprocess PCB
          8E D5 0201 440      SETIPL    #0
          FF07 31 0203 441      MOVQ     (SP)+,R4      ; Restore R4, R5
          00000008 0206 442      TSTL     (SP)+      ; Discard stacked status
          020A 443      BRW      START_OK      ; and take successful exit path
          020A 444      ;
          020A 445 20$: .LONG    IPL$_SYNCH
          020A 446      ;
          020A 447      ;
          020A 448      ;
          020A 449      ;
          020A 450      ;
          020A 451 STOP:      ;
          FE9F 30 020A 452      BSBW     CHECK_ACMODE      ; If caller does not have privilege,
          03 50 E8 020D 453      BLBS     R0,5$
          FEAB 31 0210 454      BRW      NOPRIV      ; return with error status
          00 36 A5 00 E5 0213 455 5$: BBCC    #PHDSV_PFMFLG,PHDSW_FLAGS(R5),10$ ; Turn off active monitoring
          00000065'EF 16 0218 456 10$: JSB      FLUSH_BUFFER      ; Flush outstanding buffers
          01 DD 021E 457      ; (R2, R4, R5 DESTROYED)
          02 0B A6 E8 0221 458      SETIPL    #0      ; return to IPL 0
          0223 459      PUSHL    #SS$ NORMAL      ; Assume success
          0223 460      ASSUME     PMBSV_MODE EQ 0
          0223 461      ASSUME     PMBSK_IMAGE EQ 1
          0223 462      BLBS     PMBSB_FLAGS(R6),PURGE_EXIT ; If subprocess mode,
```



```
06 11 0227 463 BRB STOP_PROCESS ; do orderly shutdown of subprocess
      0229 464 PURGE_EXIT: ; (Assumes IPL=0, (SP) = status)
50 BED0 0229 465 POPL R0
      022C 466 PURGE_EXIT R0:
3C 10 022C 467 BSBB PFM$PURGE ; Return process buffers to system
      04 022E 468 RET ; Return w/status in R0
      022F 469
      022F 470 :
      022F 471 : Orderly shutdown of subprocess
      022F 472 :
      022F 473
      022F 474 STOP_PROCESS:
      022F 475 $FORCEX_S PIDADR=PMB$E_PID(R6) ; force subprocess to exit
1D 50 E9 023D 476 BLBC R0,DASSGN_EXIT ; don't wait for message if force failed
      0240 477 $QIOW_S CHAN=PMB$W_MBXCHN(R6),FUNC=#IOS_READVBLK,-
      0240 478 P1=PMB$E_PID(R6),- ; buffer four bytes of termination
      0240 479 P2=#4 ; ...message in unneeded PID slot
      025D 480 DASSGN_EXIT:
      025D 481 $DASSGN_S CHAN=PMB$W_MBXCHN(R6) ; release mailbox
BF 11 0268 482 BRB PURGE_EXIT
      026A 483
```



```
026A 485 .SBTTL PFM$PURGE - Return all process buffers to pool
026A 486
026A 487 :++
026A 488
026A 489 : FUNCTIONAL DESCRIPTION:
026A 490
026A 491 : All data buffers and the control block are returned to pool.
026A 492
026A 493 : CALLING SEQUENCE:
026A 494
026A 495 : BSB/JSB PFM$PURGE
026A 496
026A 497 : INPUTS:
026A 498
026A 499 : none
026A 500
026A 501 : IMPLICIT INPUTS:
026A 502
026A 503 : PCB Address
026A 504
026A 505 : OUTPUTS:
026A 506
026A 507 : none
026A 508
026A 509 : IMPLICIT OUTPUTS:
026A 510
026A 511 : none
026A 512
026A 513 : SIDE EFFECTS:
026A 514
026A 515 : none - all registers preserved
026A 516
026A 517 : ROUTINE VALUE:
026A 518
026A 519 : none
026A 520
026A 521 : ENVIRONMENT:
026A 522
026A 523 : Kernel mode, IPL <= IPL$_ASTDEL
026A 524 :--
026A 525
026A 526
026A 527 PFM$PURGE::
54 007F 8F BB 026A 528 PUSHR #^M<R0,R1,R2,R3,R4,R5,R6> ; save registers
00000000'EF D0 026A 529 MOVL L^SCH$GL_CURPCB,R4 ; get our PCB address
56 011C C4 D0 026A 530 DSBINT #IPL$_ASTDEL ; protect manipulation of PMB
011C C4 D4 027B 531 MOVL PCB$_PMB(R4),R6 ; get PMB address (if any)
56 D5 0284 532 CLRL PCB$_PMB(R4) ; clear PMB pointer
37 13 0286 533 TSTL R6 ; was there a PMB
02 0B A6 02 E5 0288 534 BEQL 40$ ; exit if not
47 10 028D 535 BBCC #PMB$V_QAST,PMB$B_FLAGS(R6),5$ ; Is ACB enqueued on PCB ?
66 D5 028F 536 BSBB DEQUEUE ACB ; Yes, remove it.
50 04 A6 D0 0291 537 5$: TSTL PMB$_CORBUF(R6) ; check if current buffer there
2E 10 0293 538 BEQL 10$ ; branch if none there
50 14 B6 0F 0297 539 MOVL PMB$_BUFBASE(R6),R0 ; else set buffer address
0299 540 BSBB 50$ ; return to system
541 10$: REMQUE @PMB$_HDR(R6),R0 ; remove buffer from queue
```



```
04 1D 029D 542 BVS 20$ ; exit if none there
26 10 029F 543 BSBB 50$ ; return buffer
F6 11 02A1 544 BRB 10$ ; go back for more buffers
50 1C B6 0F 02A3 545 20$: REMQUE @PMB$Q_SBP HDR(R6),R0 ; check that subprocess queue is clear
04 1D 02A7 546 BVS 30$ ; exit if no entry
1C 10 02A9 547 BSBB 50$ ; else return buffer to system pool
F6 11 02AB 548 BRB 20$ ; check for any more
50 56 D0 02AD 549 30$: MOVL R6,R0 ; get back PMB block address
15 10 02B0 550 BSBB 50$ ; deallocate control block
50 0080 C4 D0 02B2 551 MOVL PCBSL JIB(R4),R0 ; Get JIB address
20 A0 00000A7C 8F C0 02B7 552 ADDL2 #QUOTA_CHARGE,JIB$ _BYTCNT(R0) ; Return quota to process
007F 8F BA 02BF 553 40$: ENBINT ; restore IPL
05 05 02C2 554 POPR #^M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
05 05 02C6 555 RSB ; return to caller
05 05 02C7 556
05 05 02C7 557 ;
05 05 02C7 558 ; Return buffer to pool.
05 05 02C7 559 ;
05 05 02C7 560 ;
0A A0 B4 02C7 561 50$: CLRW PMB$W_SIZE+2(R0) ; Clear type field for EXE$DEANONPAGED
00000000'EF 16 02CA 562 JSB EXE$DEANONPAGED ; deallocate memory
05 05 02D0 563 RSB
05 05 02D1 564
05 05 02D1 565 ;
05 05 02D1 566 ; The sole purpose of this routine is to mark the ACB as not queued.
05 05 02D1 567 ; It is called as a piggy-back kernel AST routine to effectively interlock the
05 05 02D1 568 ; operation with the monitoring termination code, which will conditionally
05 05 02D1 569 ; remove the ACB from the PCB.
05 05 02D1 570 ;
05 05 02D1 571 ; Inputs: R5 = ACB address, IPL = IPL$ _ASTDEL
05 05 02D1 572 ; Outputs: PMB$V_QAST cleared, all registers preserved
05 05 02D1 573 ;
05 05 02D1 574 ;
E7 A5 04 8A 02D1 575 PFM_PKAST:
05 05 02D1 576 BICB2 #PMB$M_QAST,PMB$B_FLAGS-PMB$L_ASTQFL(R5) ; Show ACB dequeued
05 05 02D5 577 RSB
05 05 02D6 578
05 05 02D6 579 ;
05 05 02D6 580 ; This routine is called from PFM$PURGE to dequeue the ACB
05 05 02D6 581 ; if PMB$V_QAST in PMB$B_FLAGS was set.
05 05 02D6 582 ;
05 05 02D6 583 ; Inputs: R6 = PMB address, IPL=IPL$ _ASTDEL,
05 05 02D6 584 ; PMB$V_QAST checked and cleared by caller
05 05 02D6 585 ; Outputs: ACB dequeued, R5 destroyed
05 05 02D6 586 ;
05 05 02D6 587 ;
55 24 A6 9E 02D6 588 DEQUEUE_ACB:
00000000'EF 16 02DA 589 MOVB PMB$L_ASTQFL(R6),R5 ; R5 <= address of embedded ACB
05 05 02DE 590 SETIPL B^10$
05 05 02E4 591 JSB SCH$REMOVACB ; Remove ACB from PCB
05 05 02E7 592 SETIPL #IPL$ _ASTDEL
05 05 02E8 593 RSB
00000008 05 02E8 594 10$: .LONG IPL$ _SYNCH
05 05 02EC 595
```



```
02EC 597      .SBTTL PFMS$GETBUF - Return PFM Buffer to Caller
02EC 598
02EC 599      ;++
02EC 600
02EC 601      : FUNCTIONAL DESCRIPTION:
02EC 602
02EC 603      : Returns a filled PFM buffer to the caller.
02EC 604      : For efficiency, this routine and FLUSH_BUFFER assume
02EC 605      : that once the collection process is awakened, it will
02EC 606      : continue calling PFMS$GETBUF until SSS_NODATA is returned, or
02EC 607      : an error is encountered.
02EC 608
02EC 609      : CALLING SEQUENCE:
02EC 610
02EC 611      : BSB/JSB PFMS$GETBUF
02EC 612
02EC 613      : INPUTS:
02EC 614
02EC 615      : R1 = Buffer address
02EC 616      : R2 = Buffer size
02EC 617      : (Should be = PFB$$_USER_BUFFER, with which it is minimized)
02EC 618
02EC 619      : NB: No checks are made for buffer accessibility.
02EC 620
02EC 621      : IMPLICIT INPUTS:
02EC 622
02EC 623      : SCH$GL_CURPCB - PCB address of current process
02EC 624      : PCB$L_PMB in PCB - pointer to PMB
02EC 625
02EC 626      : OUTPUTS:
02EC 627
02EC 628      : If R0=SS$_NORMAL or SSS$_BUFFEROVF, buffer is filled with page fault data
02EC 629
02EC 630      : IMPLICIT OUTPUTS:
02EC 631
02EC 632      : none
02EC 633
02EC 634      : ROUTINE VALUE:
02EC 635
02EC 636      : R0 = Status
02EC 637
02EC 638      : SSS$_NORMAL
02EC 639      : SSS$_NODATA
02EC 640      : SSS$_ILLSEQOP
02EC 641      : SSS$_BUFFEROVF
02EC 642
02EC 643      : SIDE EFFECTS:
02EC 644
02EC 645      : R1-R5 destroyed
02EC 646
02EC 647      : ENVIRONMENT:
02EC 648
02EC 649      : Kernel mode, IPL 0
02EC 650      :--
02EC 651
02EC 652      : NB: This code assumes that IPL synchronization (above ASTDEL) is
02EC 653      : not required because
```



```
02EC 654 : a) a command/response protocol is used between the
02EC 655 : main and sub processes
02EC 656 : b) SYSRUNDWN will unconditionally call SETPFM to
02EC 657 : turn off monitoring (i.e., assumption a) cannot be breached)
02EC 658 : c) no code outside of this module accesses PFM data structures
02EC 659 :
02EC 660 PFMSGETBUF::
54 00000000'EF D0 02EC 661 MOVL L*SCH$GL CURPCB,R4 ; Get PCB address for this process
54 011C C4 D0 02F3 662 MOVL PCB$L_PMB(R4),R4 ; Get PMB address
3A 13 02F8 663 BEQL 50$ ; BR if none
55 1C B4 OF 02FA 664 10$: SETIPL #IPL$ ASTDEL ; Protect buffer handling
26 1D 0301 665 REMQUE @PMB$Q_SBP HDR(R4),R5 ; Dequeue filled buffer
50 50 01 3C 0303 666 BVS 40$ ; Exit if none
53 0200 8F 3C 0306 667 MOVZWL #SS$ NORMAL,R0 ; Assume adequate buffer size
53 52 D1 030B 668 MOVZWL #PFB$S_USER_BUFFER,R3 ; and set number of bytes to move
08 18 030E 669 CMPL R2,R3 ; Is buffer big enough?
50 0601 8F 3C 0310 670 BGEQ 15$ ; BR if yes
53 52 D0 0315 671 MOVZWL #SS$ BUFFEROVF,R0 ; Indicate data loss
31 BB 0318 672 MOVL R2,R3 ; Adjust number of bytes to move
61 0C A5 53 28 031A 673 15$: PUSHR #^M<R0,R4,R5> ; Save regs and status
31 BA 031F 674 MOV C3 R3,PFB$B_USER_BUFFER(R5),(R1) ; Move buffer
14 A4 65 0E 0321 675 POPR #^M<R0,R4,R5> ; Restore regs and status
05 0325 676 INSQUE (R5),PMB$Q_HDR(R4) ; Return buffer to free list
0328 677 20$: SETIPL #0 ; Restore IPL
0329 678 30$: RSB ; Return to caller
50 01AC 8F 3C 0329 679
0B A4 02 8A 032E 680 40$: MOVZWL #SS$ NODATA,R0
F1 11 0332 681 BICB2 #PMB$M_ASTIP,PMB$B_FLAGS(R4) ; Show no AST in progress
50 02DC 8F 3C 0334 682 BRB 20$
ED 11 0339 683 50$: MOVZWL #SS$ _ILLSEQOP,R0
033B 684 BRB 30$
033B 685
```

```
033B 687 .SBTTL ALLPMB - Allocate PMB control block and data buffers
033B 688
033B 689 :++
033B 690 :
033B 691 : FUNCTIONAL DESCRIPTION:
033B 692 :
033B 693 :     Allocates all process structures needed for page fault monitoring.
033B 694 :
033B 695 : CALLING SEQUENCE:
033B 696 :
033B 697 :     JSB/BSB ALLPMB
033B 698 :
033B 699 : INPUTS:
033B 700 :
033B 701 :     R4     PCB address
033B 702 :     R5     PHD address
033B 703 :
033B 704 :
033B 705 : IMPLICIT INPUTS:
033B 706 :
033B 707 :     none
033B 708 :
033B 709 : OUTPUTS:
033B 710 :
033B 711 :     R6 = PMB address
033B 712 :
033B 713 : IMPLICIT OUTPUTS:
033B 714 :
033B 715 :     PCB$L_PMB contains PMB address
033B 716 :
033B 717 : ROUTINE VALUE:
033B 718 :
033B 719 :     R0 = SS$ NORMAL
033B 720 :         = Pool allocation error (SS$_INSFMEM, etc.)
033B 721 :
033B 722 : SIDE EFFECTS:
033B 723 :
033B 724 :     R0,R1,R2,R3,R8,R9 destroyed
033B 725 :
033B 726 : ENVIRONMENT:
033B 727 :
033B 728 :     Kernel mode, IPL = IPL$_ASTDEL
033B 729 : --
033B 730 :
033B 731 ALLPMB:
033B 732     1C DD     PUSHL     #SS$_EXQUOTA           ; stack no quota error code
033B 733
033B 734 :
033B 735 :     Process should have enough quota for data buffers and PMB
033B 736 :
033B 737 :
033B 738     20 AB     58     0080 C4     D0     MOVL     PCB$L_JIB(R4),R8           ; get JIB address
033B 739     00000A7C 8F     D1     CMPL     #QUOTA_CHARGE,JIB$L_BYTCNT(R8) ; quota left?
033B 740     6F     1A     BGTRU    20$           ; error if not
033B 741
033B 742 :
033B 743 :     Allocate PMB control block and insert address in PMB list
```



```

        034C 744 ;
        034C 745 ;
        6E 0124 8F 3C 034C 746
        51 0040 8F 3C 0351 747
        00000000'EF 16 0356 748
        5C 50 E9 035C 749
20 A8 00000A7C 8F C2 035F 750
        56 52 D0 0367 751
        011C C4 56 D0 036A 752
        036F 753
        036F 754
        036F 755
        036F 756
        036F 757
        036F 758
        036F 759
        0A A6 46 8F 90 036F 760
        0374 761
        0374 762
        0374 763
        0374 764
        50 14 A6 9E 0374 765
        60 60 DE 0378 766
        80 80 DE 037B 767
        60 60 DE 037E 768
        80 80 DE 0381 769
        60 60 DE 0384 770
        60 80 DE 0387 771
        038A 772
        038A 773
        66 7C 038A 774
        038C 775
        0B A6 94 038C 776
        10 A6 D4 038F 777
30 A6 60 A4 D0 0392 778
        0C A6 01 CE 0397 779
        039B 780
        59 05 3C 039B 781
        51 020C 8F 3C 039E 782 10$:
        00000000'EF 16 03A3 783
        0F 50 E9 03A9 784
        0A A2 47 8F 90 03AC 785
        14 A6 62 0E 03B1 786
        E6 59 F5 03B5 787
        03B8 788
        6E 01 3C 03B8 789
        50 8ED0 03BB 790 20$:
        05 03BE 791
        03BF 792

        MOVZWL #SS$ _INSFMEM,(SP) ; assume memory not available
        MOVZWL #PMB$_LENGTH,R1 ; get length of PMB block to allocate
        JSB EX$ALLOCBUF ; allocate control block
        BLBC R0,20$ ; check that memory available
        SUBL2 #QUOTA_CHARGE,JIB$_BYTCNT(R8) ; adjust quota
        MOVL R2,R6 ; copy PMB block address
        MOVL R6,PCB$_PMB(R4) ; insert PMB address in PCB
        ; N.B.: PFM$PURGE assumes that quota has
        ; been charged if PCB$_PMB <> 0

        ; Initialize PMB and allocate and queue data buffers.

        MOVB #DYN$_PMB,PMB$_TYPE(R6) ; Set structure type to PMB
        ASSUME PMB$_SBPHDR EQ PMB$_HDR+8
        ASSUME PMB$_ASTQFL EQ PMB$_SBPHDR+8
        MOVAB PMB$_HDR(R6),R0 ; get queue header address
        MOVAL (R0),R0 ; init empty queue flink
        MOVAL (R0)+,(R0)+ ; init empty queue blink
        MOVAL (R0),R0 ; init subprocess queue flink
        MOVAL (R0)+,(R0)+ ; init subprocess queue blink
        MOVAL (R0),R0 ; init AST queue flink
        MOVAL (R0)+,(R0) ; init AST queue blink

        ASSUME PMB$_BUFBASE EQ PMB$_CURBUF+4
        CLRQ PMB$_CURBUF(R6) ; note no current buffer

        CLRB PMB$_FLAGS(R6) ; insure that all flags are clear
        CLRL PMB$_OVERFLOW(R6) ; indicate no overflows
        MOVL PCB$_PID(R4),PMB$_PID(R6) ; Insert PID into PMB/ACB
        MNEGL #1,PMB$_LASTCPU(R6) ; force timestamping of 1st record

        MOVZWL #BUFCNT,R9 ; number of data buffers to allocate
        MOVZWL #BUFSIZ,R1 ; get size of buffer to allocate
        JSB EX$ALLOCBUF ; allocate buffer
        BLBC R0,20$ ; take error path if no memory available
        MOVB #DYN$_PFB,PFB$_TYPE(R2) ; Set structure type to PFB
        INSQUE (R2),PMB$_HDR(R6) ; queue on empty buffer list
        SOBGTR R9,10$ ; back for more buffers

        MOVZWL #SS$_NORMAL,(SP) ; Indicate success
        POPL R0 ; Return w/status in R0
        RSB
```

```

03BF 794 .SBTTL PFMS$MON - Resident Monitoring Code
03BF 795
03BF 796 :++
03BF 797
03BF 798 : FUNCTIONAL DESCRIPTION:
03BF 799
03BF 800 : Resident code called by memory management to record
03BF 801 : page fault PC and VA. Data is inserted into a buffer.
03BF 802 : When the buffer is full, it is queued for a cooperating
03BF 803 : process which outputs the data to disk.
03BF 804
03BF 805 : CALLING SEQUENCE:
03BF 806
03BF 807 : BSB/JSB PFMS$MON
03BF 808
03BF 809 : INPUTS:
03BF 810
03BF 811 : R4 = PCB address
03BF 812 : R5 = PHD address
03BF 813
03BF 814 : IMPLICIT INPUTS:
03BF 815
03BF 816 : none
03BF 817
03BF 818 : OUTPUTS:
03BF 819
03BF 820 : none
03BF 821
03BF 822 : IMPLICIT OUTPUTS:
03BF 823
03BF 824 : none
03BF 825
03BF 826 : ROUTINE VALUE:
03BF 827
03BF 828 : none
03BF 829
03BF 830 : SIDE EFFECTS:
03BF 831
03BF 832 : none
03BF 833
03BF 834 : ENVIRONMENT:
03BF 835
03BF 836 : Kernel mode, IPL = IPL$_SYNCH
03BF 837 :--
03BF 838
03BF 839
00000000 840 .PSECT AEXENONPAGED, LONG
0000 841
0000 842
0000 843 : .ENABLE LSB
02 0C A2 D1 0000 843 5$: CMPL PFB$$_RECCNT(R2), #2 : Space for time stamp and PC/VA pair ?
0C 0C 1E 0004 844 BGEQU 6$ : BR if space in buffer
55 DD 0006 845 PUSH R5 : Save PHD address
61 10 0008 846 BSB FLUSH_BUFFER_INT : Flush buffer (R2, R4, R5 DESTROYED)
55 8ED0 000A 847 POPL R5 : Restore PHD address
0093 30 000D 848 BSBW GETBUF : try to get next buffer
49 1D 0010 849 BVS 30$ : exit if none there, lose data
81 01 CE 0012 850 6$: MNEGL #1, (R1)+ : Add time stamp to buffer

```

Ini
Com
Pas
Sym
Pas
Sym
Pse
Cro
Ass


```

      81 38 A5 D0 0015 851 SETIPL #IPL$-HWCLK ; Synchronize with CPU time updating
      38 A5 D0 0018 852 MOVL PHD$$_CPUTIM(R5),(R1)+
      38 A5 D0 001C 853 MOVL PHD$$_CPUTIM(R5)- ; Update reference CPU time
      3C A6 001F 854 PMB$$_LASTCPU(R6)
      0C A2 D7 0021 855 SETIPL #IPL$-SYNCH ; Lower IPL to entry IPL
      1C 11 0024 856 DECL PFB$$_RECCNT(R2) ; Adjust record count for time stamp
      0027 857 BRB 15$ ; Rejoin mainline code
      0029 858
      0029 859 .ALIGN LONG
      002C 860 PFMSMON::
      002C 861 PUSHR #*M<R0,R1,R2,R3,R4,R5,R6> ; save registers
      011C C4 D0 0030 862 MOVL PCB$$_PMB(R4),R6 ; get PMB address
      0035 863
      0035 864 ASSUME PMB$$_BUFBASE EQ PMB$$_CURBUF+4
      0035 865
      51 66 7D 0035 866 MOVQ PMB$$_CURBUF(R6),R1 ; get buffer address & base
      04 12 0038 867 BNEQ 10$ ; branch if buffer exists
      67 10 003A 868 BSBB GETBUF ; try to get next buffer
      1D 1D 003C 869 BVS 30$ ; exit if none there, lose data
      0C A6 38 A5 D1 003E 870 10$: CMLP PHD$$_CPUTIM(R5),PMB$$_LASTCPU(R6) ; Need a CPU time stamp ?
      BB 12 0043 871 BNEQ 5$ ; BR if yes
      81 40 AE D0 0045 872 15$: MOVL FAULTPC(SP),(R1)+ ; insert pc of instruction
      81 3C AE D0 0049 873 MOVL FAULTVA(SP),(R1)+ ; insert va which faulted
      66 51 D0 004D 874 MOVL R1,PMB$$_CURBUF(R6) ; save current buffer address
      02 0C A2 F5 0050 875 SOBGTR PFB$$_RECCNT(R2),20$ ; Buffer full ?
      15 10 0054 876 BSBB FLUSH_BUFFER_INT ; Yes, flush it
      0056 877 ; (R2, R4, R5 DESTROYED)
      007F 8F BA 0056 878 20$: POPR #*M<R0,R1,R2,R3,R4,R5,R6> ; restore registers
      05 005A 879 RSB
      10 A6 D6 005B 880 30$: INCL PMB$$_OVERFLOW(R6) ; Count an overflow
      F6 11 005E 881 BRB 20$
      0060 882 .DISABLE LSB
      0060 883
      0060 884 .ENABLE LSB
      0060 885
      0C A2 3F D0 0060 886 5$: MOVL #MAXREC,PFB$$_RECCNT(R2); Re-initialize records remaining
      05 0064 887 RSB ; and return
      0065 888
      0065 889 ; (R2, R4, R5 DESTROYED)
      0065 890 FLUSH_BUFFER: ; Entry with IPL=IPL$ ASTDEL
      52 04 A6 D0 0065 891 MOVL PMB$$_BUFBASE(R6),R2 ; Get current buffer base address
      25 13 0069 892 BEQL 20$ ; Just exit if no current buffer
      3F 0C A2 C3 006B 893 FLUSH_BUFFER_INT: ; Entry with IPL=IPL$ SYNCH, R2 set
      0C A2 006B 894 SUBL3 PFB$$_RECCNT(R2),#MAXREC ; Convert records remaining to
      ED 13 006F 895 PFB$$_RECCNT(R2) ; number of records in buffer
      0071 896 BEQL 5$ ; BR if no records in buffer
      0073 897 ASSUME PFB$$_FLINK EQ 0
      20 B6 62 0E 0073 898 INSQUE (R2),PMB$$_SBPHDR+4(R6) ; insert buffer at end of write queue
      0077 899 ASSUME PMB$$_MODE EQ 0
      0077 900 ASSUME PMB$$_IMAGE EQ 1
      16 0B A6 E9 0077 901 BLBC PMB$$_FLAGS(R6),30$ ; BR if subprocess mode
      0E 0B A6 01 E2 007B 902 BBSS #PMB$$_ASTIP,PMB$$_FLAGS(R6),10$ ; If AST in progress, return
      0B A6 04 88 0080 903 BISB2 #PMB$$_QAST,PMB$$_FLAGS(R6) ; Show embedded ACB queued on PCB
      52 01 9A 0084 904 MOVZBL #PRI$_IOCOM,R2 ; R2 <= priority increment class
      55 24 A6 DE 0087 905 MOVAL PMB$$_ASTQFL(R6),R5 ; R5 <= address of ACB
      FF72 30 008B 906 BSBW SCH$QAST ; Signal full buffer available
      66 7C 008E 907 10$: CLRQ PMB$$_CURBUF(R6) ; Note that no current buffer exists
```



```
05 0090 908 20$: RSB
0091 909
51 30 A6 D0 0091 910 30$: MOVL PMBSL_PID(R6),R1 ; get params to wake up other process
0095 911 DSBINT #IPL$-SYNCH
FF62' 30 009B 912 BSBW SCH$WAKE ; and wake the process
009E 913 ENBINT
EB 11 00A1 914 BRB 10$
00A3 915
00A3 916 .DISABLE LSB
00A3 917
00A3 918 GETBUF: ; get next buffer from queue
52 14 B6 0F 00A3 919 REMQUE @PMBSQ_HDR(R6),R2 ; dequeue next buffer
1D 00A7 920 BVS 10$ ; leave with V set if none there
04 A6 52 D0 00A9 921 MOVL R2,PMBSL_BUFBASE(R6) ; save buffer base address in PMB
51 14 A2 9E 00AD 922 MOVAB PFBSB_BUFFER(R2),R1 ; skip buffer overhead
0C A2 3F D0 00B1 923 MOVL #MAXREC,PFBSL_RECcnt(R2); Initialize records remaining
10 A2 10 A6 D0 00B5 924 MOVL PMBSL_OVERFLOW(R6),PFBSL_OVERFLOW(R2) ; Copy overflow count
10 A6 10 A6 D4 00BA 925 CLRL PMBSL_OVERFLOW(R6) ; and then reset it (also clears V)
00BD 926 ; (R1 & R2 point to buffer)
05 00BD 927 10$: RSB ; return with V set or clear
00BE 928
00BE 929 .END
```


SYSSETPFM
Symbol table

- SET PAGE FAULT MONITORING

L 6

16-SEP-1984 02:31:31 VAX/VMS Macro V04-00
5-SEP-1984 03:57:09 [SYS.SRC]SYSSETPFM.MAR;1

Page 20
(7)

```

$ST1 = 00000001
ACBSM_NODELETE = 00000020
ACBSM_PKAST = 00000010
ACMODE = 00000010
ALLPMB = 0000033B R 02
ASTADR = 00000008
ASTPRM = 0000000C
BUFCNT = 00000005
BUFSIZ = 0000020C
CHECK_ACMODE = 000000AC R 02
CHECK_STOP = 000000CE R 02
CTL$GC_PHD ***** X 02
DASSGN_EXIT = 0000025D R 02
DEQUEUE_ACB = 000002D6 R 02
DYN$C_PFB = 00000047
DYN$C_PMB = 00000046
EXESAB_HEXTAB ***** X 02
EXESALCOCBUF ***** X 02
EXESDEANONPAGED ***** X 02
EXESEPID_TO_IPID ***** X 02
EXESIPID_TO_PCB ***** X 02
EXESMAXACMODE ***** X 02
EXESSETPFM = 0000006A RG 02
FAULTPC = 00000040
FAULTVA = 0000003C
FILWRT = 00000043 R 02
FILWRTPRV = 00000052 R 02
FLUSH_BUFFER = 00000065 R 03
FLUSH_BUFFER_INT = 0000006B R 03
GETBUF = 000000A3 R 03
ILLSEQOP = 00000061 R 02
IOS_READVBLK = 00000031
IPL$_ASTDEL = 00000002
IPL$_HWCLK = 00000018
IPL$_SYNCH = 00000008
JIB$C_BYTCNT = 00000020
MAXREC = 0000003F
NOPRIV = 000000BE R 02
PCBSL_EPID = 00000064
PCBSL_JIB = 00000080
PCBSL_PID = 00000060
PCBSL_PMB = 0000011C
PFB$B_BUFFER = 00000014
PFB$B_TYPE = 0000000A
PFB$B_USER_BUFFER = 0000000C
PFB$C_LENGTH = 0000020C
PFB$C_FLINK = 00000000
PFB$C_OVERFLOW = 00000010
PFB$C_RECCNT = 0000000C
PFB$S_USER_BUFFER = 00000200
PFM$C_BUFCNT = 00000005 G
PFM$GETBUF = 000002EC RG 02
PFM$MON = 0000002C RG 03
PFM$PURGE = 0000026A RG 02
PFM$BUSY = 0000005A R 02
PFM$ILWRT = 00000000 R 02
PFM$FLG = 00000004

```

```

PFMQUOTA = 00000020 R 02
PFM_PKAST = 000002D1 R 02
PHD$C_CPUIM = 00000038
PHD$V_PFMFLG = 00000000
PHD$W_FLAGS = 00000036
PMB$B_ACMODE = 0000002E
PMB$B_FLAGS = 0000000B
PMB$B_RMOD = 0000002F
PMB$B_TYPE = 0000000A
PMB$C_LENGTH = 00000040
PMB$C_IMAGE = 00000001
PMB$C_AST = 00000034
PMB$C_ASTPRM = 00000038
PMB$C_ASTQFL = 00000024
PMB$C_BUFBASE = 00000004
PMB$C_CURBUF = 00000000
PMB$C_EPID = 00000034
PMB$C_KAST = 0000003C
PMB$C_LASTCPU = 0000000C
PMB$C_OVERFLOW = 00000010
PMB$C_PID = 00000030
PMB$M_ASTIP = 00000002
PMB$M_QAST = 00000004
PMB$Q_HDR = 00000014
PMB$Q_SBP HDR = 0000001C
PMB$V_ASTIP = 00000001
PMB$V_MODE = 00000000
PMB$V_QAST = 00000002
FMB$W_MBXCHN = 0000002C
PMB$W_SIZE = 00000008
PQL$_BYTLM = 00000003
PQL$_CPULM = 00000004
PQL$_FILLM = 00000006
PQL$_LISTEND = 00000000
PQL$_PGFLQUOTA = 00000007
PQL$_PRCLM = 00000008
PQL$_TQELM = 00000009
PR$ IPL = 00000012
PRIS IOCOM = 00000001
PSL$S_PVMOD = 00000002
PSL$V_PVMOD = 00000016
PURGE_EXIT = 00000229 R 02
PURGE_EXIT_R0 = 0000022C R 02
PURGE_EXIT_R0_VEC = 000000C8 R 02
PURGE_EXIT_VEC = 000000CB R 02
QUOTA_CHARGE = 00000A7C
SCH$GC_CURPCB ***** X 02
SCH$QAST ***** X 03
SCH$REMOVACB ***** X 02
SCH$WAKE ***** X 03
SET_ASTMODE = 00000116 R 02
SET_SUBPMODE = 00000130 R 02
SS$_BUFFEROVF = 00000601
SS$_EXQUOTA = 0000001C
SS$_ILLSEQOP = 000002DC
SS$_INSFMEM = 00000124
SS$_NODATA = 000001AC

```

SYS
Tab

SYSSETPFM
Symbol table

- SET PAGE FAULT MONITORING

M 6

16-SEP-1984 02:31:31 VAX/VMS Macro V04-00
5-SEP-1984 03:57:09 [SYS.SRC]SYSSETPFM.MAR;1

Page 21
(7)

SS\$NOPRIV	=	00000024		
SS\$NORMAL	=	00000001		
SS\$PFMBSY	=	00000204		
START		000000E3	R	02
START_OK		0000010D	RR	02
STOP		0000020A	RR	02
STOP_PROCESS		0000022F	RR	02
STOP_VEC		000000C5	RR	02
SUBFUNC		00000090	R	02
SY\$CREMBX	*****		GX	02
SY\$CREPRC	*****		GX	02
SY\$DASSGN	*****		GX	02
SY\$FORCEX	*****		GX	02
SY\$GETCHN	*****		GX	02
SY\$QIOW	*****		GX	02

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes													
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE			
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE			
YEXEPAGED	000003BF (959.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG			
AEXENONPAGED	000000BE (190.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG			

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.07	00:00:00.26
Command processing	106	00:00:00.55	00:00:01.73
Pass 1	407	00:00:15.76	00:00:35.84
Symbol table sort	0	00:00:02.35	00:00:05.76
Pass 2	165	00:00:03.46	00:00:08.12
Symbol table output	16	00:00:00.14	00:00:00.22
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	727	00:00:22.36	00:00:51.95

The working set limit was 1500 pages.
88818 bytes (174 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1517 non-local and 40 local symbols.
929 source lines were read in Pass 1, producing 20 object records in Pass 2.
38 pages of virtual memory were used to define 36 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-----	-----
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	13
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	32

1718 GETS were required to define 32 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LISS:SYSSETPFM/OBJ=OBJ\$:SYSSETPFM MSRC\$:SYSSETPFM/UPDATE=(ENH\$:SYSSETPFM)+EXECML\$/LIB

0388

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY